Multiagent Systems for Social Computation

Multiagent Systems for Social Computation

Michael Rovatsos
The University of Edinburgh
Edinburgh EH8 9AB
United Kingdom
mrovatso@inf.ed.ac.uk

ABSTRACT

This paper proposes *social computation*, i.e. large-scale man-machine collaboration mediated by digital interaction media, as a vision for future intelligent systems, and as a new challenge for multiagent systems research. We claim that the study of social computation suggests a re-interpretation of many traditional AI endeavours, has huge potential application benefits, and presents the field of multiagent systems with novel, exciting research questions. We introduce an abstract model of social computation that helps capture some of its core research problems more precisely. We explore the potential contribution of multiagent systems technologies to the solution of these problems by exposing the close relationship between social computation and existing methods in multiagent systems. We describe how these methods could be reused in this novel application context, what methodological implications this has, and argue that the resulting cross-fertilisation will be highly beneficial for both sides.

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Multiagent Systems

General Terms

Design, Experimentation, Human Factors

Keywords

Social computation, human-based computation, crowdsourcing, collective intelligence

1. INTRODUCTION

Massive advances in network connectivity and increased affordability of computer hardware have recently led to a flurry of web-based applications that mediate interaction within human collectives [10]. This has, in turn, led to an increased interest in "collective intelligence" [6, 8] applications, where human user input is used to gather data, derive new knowledge, execute complex computational tasks, and/or improve user experience. Many of these applications currently emphasise the crowdsourcing and human-based computation [5] aspects of collective intelligence, focusing on the way human intelligence can contribute to the solution of complex computational problems [4].

A more ambitious vision of future *social computation* (SC) applications, however, has humans and machines contributing in varying measures to systems that solve complex computational and societal problems that involve a variety of diverse stakeholders, acting as producers and consumers of parts of the overall, decentralised computation.

Examples of such potential future systems include: ride-sharing applications where algorithms support travellers in collaborative route planning while also managing congestion in the urban areas involved; healthcare systems that monitor patients and their clinical treatment plans while prioritising use of staff time and resources based on long-term data analysis; software development platforms that allow companies to outsource production to teams of freelancers, enable component integration/reuse through code analysis, and detection of social interaction patterns among developers.

These kinds of systems share a set of characteristics that set them apart from other decentralised IT systems: They embody a *multi-perspective notion of hybrid man-machine intelligence*, where the capabilities of humans and computational artefacts complement each other (rather than machines imitating human intelligence as in traditional AI), and different facets of intelligent behaviour are relevant to different stakeholders. They are *continually co-designed by programmers and end users* through human and machine contributions, be it as a consequence of emergent behavioural dynamics (e.g. variations of quality of a recommender system) or because the open-ended architecture of the Web allows re-appropriation and coupling of different existing components (e.g. through mashups). This evolutionary aspect can, in fact, be seen as part of the "intelligence" of the system, as it enables constant adaptation to changing user and designer needs and behaviour.

In this paper, we argue that SC offers a challenging new domain for the field of multiagent systems (MAS) that has huge potential scientific and application benefits. We identify a set of core research problems in the area, discuss their relationship to MAS, and highlight key methodological issues that need to be addressed if we want to apply MAS methods to SC problems.

2. ABSTRACT SOCIAL COMPUTATIONS

We start by presenting an abstract model of SCs that allows us to describe the research problems they give rise to more precisely. Assume a set of human/machine agents $A = \{1, ..., n\}$ and a set of local functions $F = \{f_1, \ldots, f_m\}$ these agents can compute (where, typically, $m \ll n$), such that $F_i \subseteq F$ are the functions of agent i. The global set of variables in the system $X = \{x_1, \ldots, x_k\}$ determines what

Appears in: *Alessio Lomuscio, Paul Scerri, Ana Bazzan, and Michael Huhns (eds.), Proceedings of the 13th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2014), May 5-9, 2014, Paris, France.*

the inputs and outputs of each function are, where every variable x_l has a domain D_l, and $X_i \subseteq X$ indicates which variables i has access to. Note that as many (often most) agents will be human users, many of the f_i will not have (known) formally precise, algorithmic representations.

Every local function $f \in F_i$, has input and output sets X^f_{in}, $X^f_{out} \subseteq X$, and we have $f : D^f_{in} \rightarrow D^f_{out}$ where $D^f_{in} = D_{i_1} \times \ldots \times D_{i_s}$ and $D^f_{out} = D_{o_1} \times \ldots \times D_{o_t}$ denote the domains of the input and output variables sets of f, i.e. $X^f_{in} = \{i_1, \ldots, i_s\}$ and $X^f_{out} = \{o_1, \ldots, o_t\}$. We will assume that agent i has access to the input variables of its local functions ($X^f_{in} \subseteq X_i$), and that it can (or is willing to) only compute the outcome of a local function for a restricted subset of the possible inputs $D^f_i \subseteq D^f_{in}$.

To overlay this collection of local functions with a network structure, we introduce a neighbourhood function $N : A \rightarrow 2^A$ which maps every agent i to a set of agents $N(i)$ that i has access to, including itself (these are nodes that can be found via a search, are acquaintances, etc). A set of discrete timesteps $T = \{t_1, t_2, \ldots\}$ is used to specify values at specific points in time, e.g. x^t_l denotes the value of variable x_l at timestep t, f^t denotes that function f is invoked at timestep t and so on (if a computation takes k timesteps, we have $f^t(x^t) = y^{t+k}$).

With this, we can define a *sequential social computation function* $SC = (f, \{F_i\}_{i \in A}, N, I, t)$ to compute f using agents A on an input set $I \subseteq D_{i_1} \times \ldots D_{i_m}$ where $\{i_1, \ldots, i_m\} \subseteq \{1, \ldots |X|\}$ as a procedure that calculates f for each $x^t \in I$ given at time t after k timesteps such that

$$f(x^t) = f_n^{t+k_1+\ldots+k_{n-1}} \circ \ldots \circ f_2^{t+k_1} \circ f_1^t(x^t) = y^{t+k}$$

and output set O as the set of values y^{t+k} that result from this computation, where the following conditions hold for every $1 \leq j \leq n-1$, and $k = \sum_{j=1}^{n} k_j$:

1. There is some agent i with $f_j \in F_i$, and $f_{j+1} \in F_{N(i)}$.
2. For any two $f_j \in F_i$ and $f_{j+1} \in F_{i'}$, $x^{t+k_1+\ldots+k_j} \in D^{f_j}_i$ and $x^{t+k_1+\ldots+k_{j+1}} \in D^{f_{j+1}}_{i'}$ if $f^t_j(x^t) = x^{t+k_j}$.

The idea behind this is fairly simple: An SC calculates a target function f that is the result of a sequential application on inputs received from predecessor functions (or from the environment – we impose no constraints on the inputs other than that they be accessible to the agent operating on them) and passed on to the subsequent computation node. Condition 1. restricts the SC to sequences that can be constructed using only neighbours of the currently executing agent in each step, i.e. the overall computation is constrained by the network structure. Condition 2., on the other hand, constrains the computation of every local function to those inputs that the respective agent can (or is prepared to) process. Figure 1 illustrates the structure of these sequential computations graphically. It is possible to extend this model to parallel, synchronised computations: For this, we need a collection $\{SC_1, \ldots, SC_m\}$ of sequential SCs, and a set of constraints defined on *synchronisation variables* $X_{sync} \subseteq X$ of the form $(i, j, t, x \; op \; x')$ where *op* specifies a relationship (e.g. $=, <, >$) that must hold between the values of variables x and x' appearing in SC_i and SC_j at time t, respectively.

Note that our model neither requires that every agent needs to be different or only involved in a single step (e.g. the sequence could involve polling different agents and then aggregating the result in a central node), that agents are heterogeneous in terms of what functions they can perform, or that all of the output variables they compute are needed by the subsequent node – some of those just effect local changes that are irrelevant for the overall computation.

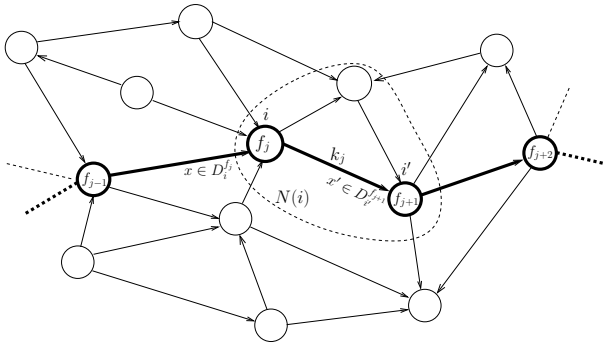

Figure 1: An abstract SC: Network edges depict neighbourhood relations N, with bold arrows for edges that are used to compute f. The highlighted agent i performs function $f_j \in F_i$ based on input x received from the previous node and, optionally, also variables locally known by i. After k_j timesteps successor i' continues the computation invoking f_{j+1}.

This model is deliberately abstract and simplified: it does not account for asynchronous processing, non-determinism, hierarchical abstraction, or aggregation. Also, it does not imply any commitment to the algorithmic representations that will be used for implementation. However, it captures the key elements of SC systems: a network structure that provides connectivity between local processes, constraints on the circumstances under which local computation will be performed, and fully decentralised information and control.

3. CORE RESEARCH PROBLEMS

At the abstract level, our definition above does not appear very different from a traditional distributed (autonomous) systems model. The challenging aspects of SC arise from the fact that humans play a significant role in these computations, significantly limiting *observability* and *predictability* of the system. We discuss several implications of this in the following exposition of a number of core research problems formulated using our model:

Synthesis *Given f, input set I, and a set of agents A with capabilities $\{F_i\}_{i \in A}$, what is a concrete sequence $f_n \circ \ldots \circ f_1$ that computes $f(I)$?* The solvability of this problem depends on the way in which the functions are represented, i.e. this question cannot be answered at the level of our above model, which may include functions that are neither machine- nor human-computable. Certainly, for many functions computed by humans, there is little hope that we can describe those using rigorous formalisation.

Verification *Does a given sequence $f_n \circ \ldots \circ f_1$ compute the target function f correctly on inputs I?* While in principle much simpler than synthesis, in many real-world domains no agent will be able to verify whether others' local functions have been (correctly) executed, e.g. when they involve spatially dispersed physical action in the environment, or when they involve genuinely non-verifiable results (opinions, expert knowledge). An important question here is how human-based verification can be used to improve the "safety" of the SC system, for example through reputation systems.

Recruitment *Given a function f, input set I, and agents A, how can we identify a set of participants $P \subseteq A$ that will compute $f(I)$?* While this could be solved through exhaustive enumeration in a system with complete information, in human-centric systems we will normally not know under

which conditions participants can/will perform the task from the outset. Also, there is a circular dependency between task specification and recruitment: How can users decide to participate before an overall description of the computation is presented to them, which would, in turn, require specifying which of them will contribute to this computation?

Incentivisation *Given special variables $X_{inc} \subseteq X$ that are under the control of an agent i, how should i choose this to solve the recruitment problem for a specific input set I?* This is a more specific sub-problem of recruitment: In our model, incentives can be viewed as variables X_{inc} whose values are set by the agent initiating an SC (e.g. modifying bank credit after task completion). How would these need to be chosen to persuade an adequate set of participants to contribute, and to execute their local tasks correctly?

Synchronisation *Given a set $\{SC_1, \ldots, SC_m\}$ of sequential SCs, what set of constraints $(i, j, t, x\ op\ x')$ will enable all of them to be executed correctly?* This essentially asks how we can resolve conflicts that could arise from the parallel execution of more than one sequential SC, and is important when we consider the open-world semantics of the Web, where one SC may not be aware of the existence of the other, but may share resources/participants with it.

Composition *Given a set $\{SC_1, \ldots, SC_m\}$ of SCs that compute $\{f_1, \ldots, f_m\}$, respectively, and a set of constraints of the form $(i, j, t, x\ op\ x')$, what function f does the overall system compute?* Complementary to synchronisation, in a sense, this question addresses more general problems of compositionality and emergent behaviour, as it may be the case that the joint effect of several SCs does not occur "by design", but only as an indirect consequence of running several of them in parallel or in sequence.

Optimisation *Given a quality measure q for SCs and an input set I, identify $SC^* = \arg\max_{SC} q(SC)$.* Since SCs usually operate in resource-constrained environments, they will have to satisfy certain optimality criteria. Different from other kinds of systems, the quality of an SC is intrinsically multi-perspective and subjective (e.g. is it *fun*?), and its overall evaluation is subject to continual change. Hard, *a priori* optimality criteria are unlikely to work here.

At the *computational* level, casting these problems in an SC context suggests a strongly incremental approach, where any successful solution method would need to specify (i) how it will discover new information over time to refine and improve an existing model; (ii) how it will adapt its operation to changing information; and, (iii) how it will expose its adaptability to designers and users that act as stakeholders in this process of evolutionary design.

At the *human factors* level, they present themselves within a social and cognitive context, raising many issues related to ethics, governance and stakeholder engagement, transparency and accountability, privacy and safety, and responsible research and innovation.

4. THE CONTRIBUTION OF MAS

At first glance, most of the existing SC systems [10] contain functional building blocks that look very familiar from a MAS point of view: *discovery* (in the broadest sense, including data gathering, preference elicitation, user profile acquisition, advertisement and solicitation), *assignment* (including network exploration, peer search, filtering, matchmaking, negotiation, and agreement), *execution* (automated and human-based sensing and monitoring, performance of algorithmic and physical tasks, tracking, repair, and recovery, conflict and dispute resolution) and *analysis* (feedback elicitation, reputation and provenance modelling, pattern recognition and data mining, predictive analytics).

This suggests that many MAS techniques should be appropriate for the study and implementation of SCs: Agent architectures, platforms, communication, and programming languages can be used to implement SCs and design the interactions occurring within them; distributed search, matchmaking, and automated negotiation methods can be used to organise assignment and task allocation; multiagent planning, distributed constraint satisfaction, teamwork, reactive monitoring and execution provide a rich arsenal of methods for execution, where coordination techniques like norms, institutions, and multiagent organisations can provide the scaffolding required to regulate social interaction; multiagent learning and trust and reputation modelling, finally, offer highly relevant analysis and adaptation techniques.

On the other hand, existing SC applications exhibit a glaring lack of theoretical foundations and solid engineering principles of the kind developed by MAS researchers. Most of these systems are still developed in an *ad hoc* fashion, and existing programming frameworks [1, 7] for them mainly provide API-level integration and scripting to simplify the application development process. Although SC is being addressed as a domain by such diverse areas of computer science as distributed computing, human-computer interaction, computer-supported cooperative work, and service-oriented computing, none of these seem to dispose of the theoretical and algorithmic underpinnings and the engineering methodologies that have been developed for open, decentralised systems within the MAS community. A striking illustration of this is that hardly any of the existing systems propose an overall design for all of the functional components described above for a specific application, let alone at a more domain-independent level.

Yet, somewhat surprisingly, the only area within MAS that has been successfully applied to SC problems so far (e.g. [3, 4]) appears to be what we could term "agreement technologies", i.e. automated negotiation and distributed decision making techniques that mostly fall into the category of game-theoretic AI [2]. With this exception, none of the aforementioned MAS sub-topics is currently used to design real-world, large-scale SC systems. Moreover, even in the case of agreement technologies, their use is strictly limited to the design and analysis of SC applications, and does not extend to their *engineering*, a fact that is striking if we consider that MAS research has been dealing with similar kinds of systems structure for the past 30 years.

5. METHODOLOGICAL IMPLICATIONS

In this section, we propose a set of methodological shifts that would be required in order to best exploit the opportunities for MAS techniques to play an important part in the study and development of future SC systems. While this does not constitute a specific research agenda *per se*, it suggests a different outlook on distributed intelligent systems that may spur new directions for MAS research:

In situ research Research into SC only makes sense if it is done *in situ*, i.e. designed, evaluated, and improved under real-world conditions. This is due to the fact that such systems are "mostly human" multiagent systems where the vast majority of agents are humans interacting with each other and the system. They continually contribute constituent human computations, while artificial agents provide certain core services like automated matchmaking and negotiation, workflow planning, execution and monitoring, and data analysis. Unlike other types of systems, this necessitates that human factors are considered in all stages of

scientific investigation, rather than, for example, consulting social scientists or usability experts only for certain aspects of the design process.

Web as multiagent platform The history of MAS research has shown that "bespoke" multiagent platforms developed by the field are computationally too heavy for massive-scale networked interaction, and conceptually too involved to achieve uptake from developers who are not experts in agent technologies. As both very large-scale participation and flexible bottom-up interoperability are essential for SC systems to succeed, future MAS in these domains must remain as close to the Web infrastructure as possible, given that the Web is the only known distributed computing infrastructure that satisfies these requirements. Following the principles of linked/open data and architectural paradigms like REST will ensure that multiagent-based SC systems can be easily combined with external data sources and will interoperate with third-party applications.

Empirical strategic analysis Appropriate use of incentives is essential to the successful organisation of any social computation, but can only be evaluated in the light of the actual response user populations exhibit toward any given incentive scheme (regardless of whether this involves benefits derived directly from the interactions, or additional incentives provided by the system designer) [4, 9]. Thus, we need to focus less on normative models of strategic behaviour that rely on *a priori* knowledge of participants' preferences and their rationality, and more on an empirical analysis of observed participant behaviour. Learning is likely to play a crucial role here to some extent, as are models from behavioural economics, but it is likely that modelling of social norms and values will play an equally important part in deriving adequate models of real, observed user behaviour.

Evolving semantics Traditional models of semantics for data, knowledge, and agent communication focus on hard, idealised properties. Their application is likely to be of limited applicability in SC systems, as the continual feedback loop between information presented to users and their response to it gives rise to a perpetual evolution of meaning. New models are needed that describe and utilise the ways in which symbols and signals are re-interpreted over time in specific contexts by changing populations of human users.

From agents to collectives SC systems often have millions of users, many of whom only interact with a system only very sporadically. This suggests that the fundamental unit of modelling and analysis should not be an individual agent, but rather a collective, which "makes individual agents available" for specific interactions and computations. While MAS research provides useful models of delegation and aggregation, *stratification* is an essential modelling technique that is largely amiss from existing research. It would help build generalised models of individuals by classifying them into different collectives based on similar motives, skills, or behaviours. Using such methods will also be necessary in order to be able to build manageable system models where, for example, large-scale social networks are never described at the level of each interacting individual, but rather as much smaller structures that distinguish only among classes of individuals and interactions.

While following any of these principles alone is not groundbreaking in itself, we believe that their combination does create a very different outlook on the way we approach the development of theoretical models, architectures, algorithms, and implemented multiagent systems. It essentially amounts to abandoning the design of agents and their interaction mechanisms as pre-specified, fairly predictable entities in favour of a view of systems that evolve through constant interaction with large, unpredictable, diverse human user populations, with significant consequences for assumptions, solution concepts, and performance criteria.

6. CONCLUSIONS

In this paper, we outlined a vision for future social computation systems that extrapolates from recent advances at the intersection of social computing and human-based computation. We argued that multiagent systems technologies can play a major role in realising this vision, and that the field has to confront significant challenges if it wants to play this role: On the one hand, fundamental assumptions and ways of working need to be rethought. On the other, untapped potentials to use techniques that have evolved in separation from the reality of the Web need to be exploited, building on the similarities between social computation and multiagent systems, which are, in our view, striking.

We are convinced that cross-fertilisation between the two fields would be highly beneficial for both sides: It would help SC development evolve from a "black art" to a solid engineering discipline on the one hand. On the other, it would allow us to use many MAS techniques to build the next generation of collectively intelligent systems, where humans and machines jointly solve problems harder than those that can be solved by current AI by orders of magnitude. We hope that this paper will help promote this kind of fruitful interaction between the two fields.

7. ACKNOWLEDGMENTS

The research presented in this paper has been funded by the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement n. 600854 "*Smart-Society – Hybrid and Diversity-Aware Collective Adaptive Systems: Where people meet machines to build smarter societies*" (http://www.smart-society-project.eu/).

8. REFERENCES

[1] S. Ahmad, A. Battle, Z. Malkani, and S. D. Kamvar. The Jabberwocky Programming Environment for Structured Social Computing. In *Procs UIST 2011*, pages 53–64, Santa Barbara, CA, 2011.

[2] E. Elkind and K. Leyton-Brown. Special issue on algorithmic game theory. *AI Magazine*, 31(4), 2010.

[3] S. Jain, Y. Chen, and D. C. Parkes. Designing incentives for online question and answer forums. In *Procs EC'09*, pages 129–138, New York, NY, 2009.

[4] M. Kearns. Experiments in social computation. *Comm ACM*, 55:56–67, 2012.

[5] E. Law and L. von Ahn. *Human Computation*. Morgan & Claypool, 2011.

[6] P. Levy. *Collective Intelligence: Mankind's Emerging World in Cyberspace*. Perseus Books, 2000.

[7] G. Little, L. B. Chilton, M. Goldman, R. C. Miller. TurKit: Human Computation Algorithms on Mechanical Turk. In *Procs UIST 2010*, 2010.

[8] T. W. Malone, R. Laubacher, and C. Dellarocas. The collective intelligence genome. *Sloan Management Review*, 51(3):21–31, 2010.

[9] N. Peled, Y. Gal, and S. Kraus. In P. Yolum, K. Tumer, P. Stone, and L. Sonenberg, editors, *Procs AAMAS 2011*, pages 345–352, Taipei, Taiwan, 2011.

[10] N. R. Shadbolt, D. Smith, E. Simperl, M. van Kleek, Y. Yang, and W. Hall. Towards a classification framework for social machines. In *Procs WWW 2013*, pages 905–912, 2013.